Effective retrieval and new indexing method for case based reasoning: Application in chemical process design

Negny Stéphane*, Riesco Hector, Le Lann Jean Marc

Université de Toulouse, Laboratoire de Génie Chimique (LGC)—CNRS 5503, INPT-ENSIACET, 4 allée Emile Monso, BP 44362, 31432 Toulouse Cedex 04, France

Keywords:
Case based reasoning
Adaptation guided retrieval
Fuzzy sets
Sphere indexing algorithm
Design

In this paper we try to improve the retrieval step for case based reasoning for preliminary design. This improvement deals with three major parts of our CBR system. First, in the preliminary design step, some uncertainties like imprecise or unknown values remain in the description of the problem, because they need a deeper analysis to be withdrawn. To deal with this issue, the faced problem description is soften with the fuzzy sets theory. Features are described with a central value, a percentage of imprecision and a relation with respect to the central value. These additional data allow us to build a domain of possible values for each attributes. With this representation, the calculation of the similarity function is impacted, thus the characteristic function is used to calculate the local similarity between two features.

Second, we focus our attention on the main goal of the retrieve step in CBR to find relevant cases for adaptation. In this second part, we discuss the assumption of similarity to find the more appropriated case. We put in highlight that in some situations this classical similarity must be improved with further knowledge to facilitate case adaptation. To avoid failure during the adaptation step, we implement a method that couples similarity measurement with adaptability one, in order to approximate the cases utility more accurately. The latter gives deeper information for the reusing of cases.

In a last part, we present a generic indexing technique for the base, and a new algorithm for the research of relevant cases in the memory. The sphere indexing algorithm is a domain independent index that has performances equivalent to the decision tree ones. But its main strength is that it puts the current problem in the center of the research area avoiding boundaries issues. All these points are discussed and exemplified through the preliminary design of a chemical engineering unit operation.

1. Introduction

The design phase of a product is a crucial phase of its life cycle. Indeed, it influences the future of the product because many important decisions taken during this phase can condition its future costs (involving engineering, production and commercial aspects), its future developments and its acceptance on a market. Therefore, the design phase was the subject of many studies coming from scientific research, industrial and normative domains. All these studies describe the design process around the following principal stages: requirements expression, preliminary design and detailed design. The differences between all the processes, are in the sub-stages encompass in these principal stages, and in the implementation of them (sequentially or simultaneously). This article is focused on the preliminary design stage and more specifically on the embodiment design. For a product or a process, this sub-stage consists in choosing the technologies used, the materials, the structural dimensions.

Currently to characterize the design phase, there is a general acceptance of the classification into three main types: routine, innovative or creative design. The key distinction between them is based on the knowledge available to satisfy the desired requirements (Chandrasekaran, 1990):

- Routine design: All the variables, their associated ranges, and the knowledge to find the variables values can be reached from existing past design (well defined space of potential design).
- Innovative design: In contrast with the aforementioned design, the ranges of the potential values for variables are enlarged. Consequently, even if the structure is familiar, the application is none common: new performances, new functionalities, etc., due to new combinations of variables and new values for them.
- Creative design: Here, new variables are introduced (and obviously their domains of variation), as a consequence it extends the solution space of potential design.

* Corresponding author.
E-mail address: stephane.negny@ensiacet.fr (N. Stéphane).

doi:10.1016/j.engappai.2010.03.005

The major difference between these design types is the level of abstraction of the knowledge available. In routine design, problems are well understood with well known decision variables, decision points, outcomes, constraints. In innovative and creative design there is an upper level of abstraction on knowledge, resulting in an incomplete knowledge on the components, constraints, domain of validity for the variables. As a result, the complexity of the reasoning process increases and requires iterations in order to incrementally decrease the level of abstraction to reach a solution. For both innovative and creative design (Cortes Robles et al., 2009) propose a method to reduce or to avoid (depending in the problem complexity) the iterative process.

In routine design, the well defined knowledge offers the possibility to develop tools to support human designers and to automate some design tasks. To achieve these goals, we need to manage the knowledge developed during the embodiment design sub-stages. Knowledge Management (KM) encompasses a range of methods and techniques in order to identify, represent, organize, create, memorise and distribute the necessary knowledge in an organization. The challenge is to develop a computer aided design tool to support the design process to produce a better design in a shorter time horizon. The reuse of well known and optimized past design experiences increases the quality of the solution, decreases the time of the design phase and improves the efficiency of the process. There are several approaches that have been developed to handle knowledge Management during routine design. Among them, the three major approaches currently used are: rule based (RB), constraint satisfaction problems (CSP) and case based reasoning (CBR) (some additional approaches like; prototype based reasoning, axiomatic design (Suh, 1990), procedural approach, model based approach are also used).

In RB, the knowledge is represented with rules of the form IF X THEN Y. Where X is a condition and Y the action (X can be a composite condition). Furthermore, a rule based system has an inference engine to determine which rules to fire. The first drawback is the time consuming aspect of the knowledge acquisition task. Acquiring domain specific information and converting it into some rules is a huge work especially with problems where some shadow areas remain. The second major drawback concerns the maintenance of the whole knowledge. The number of rules grows up sharply, besides they also change, leading to problems of rules management (for example, sometimes two rules can result into two opposite actions).

In CSP, the knowledge related to a problem is segmented in elementary pieces, modelised by constraints: logical expressions, mathematical equalities or inequalities, range of validity. Based on these pieces, a knowledge model and reasoning are built. When a new problem is faced, it is submitted to the knowledge model, then the reasoning is driven through the constraints in order to reduce the domain of the possible values for the variables. This approach has two principal strengths; its ability to reach new solutions and to establish that a problem has no solution (over constraint problem for example). Here again the major drawback is the huge work dedicated to the extraction, interpretation and formulation of the domain knowledge.

The CBR approach, based on the human reasoning, try to propose a solution to a current problem by establishing some similarities with problems previously solved (i.e. cases) and stored in a memory (case base). The main principle of CBR is: *similar problems have similar solutions*. Compare to both previous approaches, it requires significantly less knowledge extraction, the principal relevant characteristics of the problem and its associated solution are sufficient. Whereas this approach has a learning step to extend the number of cases in the memory, it needs to gather an important number of cases in order to widely cover the problem and solution spaces and to be effective, especially during the CBR start up phase. Because of its many advantages, this approach is retained for this work. Among them, we can underline: its reduced knowledge acquisition task, its flexibility in knowledge modelling, its ability to learn, its possibility for reasoning with incomplete or imprecise data, and its rapidity to create and to maintain a computer decision support tool for designers.

Even if it is commonly accepted that CBR came from cognitive science research on human dynamic memory (Schank, 1982), foundations of CBR can be searched in different additional disciplines: knowledge representation, machine learning and mathematics (Richter and Aamodt, 2006). In CBR, the central notion is a case which is a contextualised piece of knowledge representing a previous experience that can be structured in accordance with the CBR purpose. There have been various models to represent the CBR method (Hunt, 1995; Allen, 1994; Kolodner, 1993). Currently, there is a general acceptance of R^4 model introduced by Aamodt and Plaza (1994) extended by Finnie and Sun (2003) with the well known steps: represent, retrieve, reuse, revise and retain, Fig. 1. This R^5 model is more complex and deeper than this mere presentation because each step involves a number of more specific sub-processes with their own difficulties (Pal and Shiu, 2004).

The goal of this paper is to present a CBR system for the embodiment design dedicated to chemical engineering unit operations. In this paper, the attention is focused on the two first steps because they are crucial steps for the success of CBR systems. During embodiment design, some characteristics of a system are not clearly defined or not yet fixed; consequently some uncertainties remain on them. These uncertainties must be taken into account in the problem description and in the research of similar past experiences. Here we introduce the fuzzy sets theory to soften the problem description and to model uncertainties. Moreover to avoid a prohibitive processing time during the retrieved steps, the case base is often indexed in order to restrain the similarity calculation to a subset of the most relevant source cases. In most of the CBR systems, a decision tree index method is implemented to extract this subset. In this article, we propose a new generic method for the case base indexation coupled with a specific research algorithm.

The rest of the paper is articulated as follows. In the next section, we present the way to calculate similarity with the fuzzy sets theory with our case representation. We also motivate the need to link retrieval and adaptation and we introduce a method

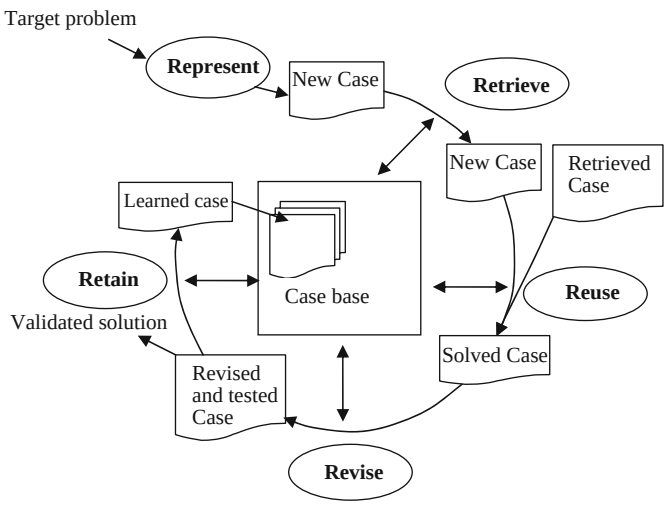

Fig. 1. R^5 CBR cycle.

for adaptation-guided retrieval. Section 3 focuses on the case base organization. Then the algorithm to retrieve source cases is described. Before to conclude, in Section 4 the similarity measurements with the fuzzy sets theory and the new research algorithm are tested through an example in preliminary design.

2. Relevant cases selection

2.1. Engineering domain of application

Chemical engineering is the part of the engineering which deals with processes that convert raw materials (more particularly chemical compounds) into more useful or/and valuable products through several transformations, under economical, environmental, safety, energy constraints. A chemical process can be decomposed into individual sub-processes called unit operations: chemical reactors, separators, mixers, heat exchangers. While the example presented here is dedicated to one unit operation i.e. distillation (it is extended to absorption in the same formulation), the method implemented can be easily extended to the other ones.

Distillation is a technique for the separation of chemical compounds based on the differences in their volatilities in a boiling liquid mixture. One of the industrial apparatus for distillation is columns with packing inside (called packed columns). Schematically, a packed column is a hollow tube filled with a packing material. In fact, it is more complex because it has inlet(s), outlet(s), a heating system, a cooling system, a reflux system, distribution systems. The packing can be randomly done with small objects (random packing) or specifically designed; (structured packing), Fig. 2. The packing has a central role in industrial distillation, because its purpose is to improve contact between the gas phase and the liquid phase, and consequently it affects the purity of the compounds in the outlet streams, the distillation column design and dimensions. Currently, there are many types of random and structured packing, for example more than 30 types of random packing with different shapes. Moreover, each type of packing is available in numerous sizes and materials.

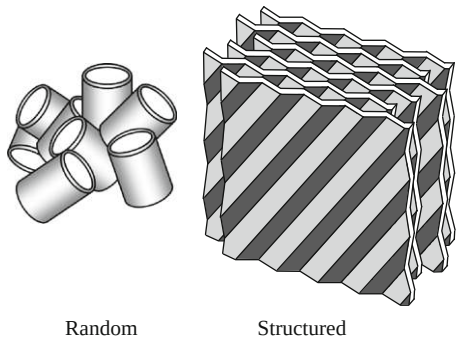

Fig. 2. Examples of column packing.

The CBR system presented here is a design support system for the choice and the design of packing for separation columns.

Because of its level of generality, the CBR method is used in various domains and activities. In chemical engineering, few studies are dealing with preliminary design. This step is often based on knowledge and past experiences of experts because of the complexity of the chemical and physical phenomenon that occur. But this step is crucial for the remainder of the design; it gives an initial guess for the future solution. In an industrial context seeking to reduce the time during the whole design process, an effective tool dedicated to preliminary design will be helpful, because a good preliminary design allows a saving of time thereafter. In this context, there is a need for new methods (and a tools) to capitalize experts knowledge to propose rapidly a high quality solution. Currently in our domain, the model based approach is widely used. But recently CBR has found some applications: (Surma and Braunschweig, 1996) for flowsheets construction, (Seuranen et al., 2005) for separation process, (Lopez-Arevalo et al., 2007) for the generation of process alternatives, (King et al., 1999) for minimizing environmental impact of separations, (Avramenko et al., 2004) and (Avramenko and Kraslawski, 2005) for a decision support system for reactive distillation, (Kraslawski et al., 1995) for the selection of mixing equipment.

2.2. Cases representation

As mentioned earlier, a case can be described to be a contextualised piece of knowledge representing a previous experience. The information encoded about the past experiences, depends on the domain of application as well as on the goal for which the cases are used. The main goal of the case representation is to traduce the knowledge needed in a relevant way: to identify the main characteristics or ontologies to describe a problem and its associated solution, and to ensure the retrieval of the most appropriated case(s) (Bergman et al., 2006). Of course, experts point of views and experiences are often mandatory to extract and represent the relevant knowledge. To support the choice of the representational format, (Pal and Shiu, 2004) give a list of several factors to consider: the internal structure of cases, the language or shell chosen, the indexing and search mechanism planned. They also compare the traditional data modelling approach according various criteria summarized in Table 1. For our proposal the cases are structured with attributes values pairs (a comment part is also added to give deeper feedback information like success or failure, suggestion of solution implementation), and represented with the object oriented representation.

With the packing design purpose, the features of the problem description and its associated solution are summed up in Table 2. Regardless of what a case represents, we can underline that the features have to be filled with different data format: textual (for mixture in the problem description, for materials and type of packing for the solution) and numeric (for the others). Solutions are described with the same global structure but some differences

Table 1
Comparison of traditional case representations (Pal and Shiu, 2004).

	Relational approach	Object oriented approach	Predicate logic approach
Compactness	Medium	High	Low
Application independency	Yes	No	No
Software reusability	No	Yes	No
Case Base scale	Large	Large	Small
Retrieving feature values for computation	Easy	Easy	Difficult
Case organization method	Keys	Inheritance/reference	Data definition

Table 2
Case representation.

Problem features	Solution features
Mixture	Type of packing
Pressure	Material
Temperature	Specific area
Inlet flow rate	Geometrical characteristics
Reflux	

can be noticed when the target solutions are detailed. For example geometrical characteristics depend on the type of packing, therefore when this feature is detailed the characteristics described and their number are different from one type of packing to another.

In the problem description, the feature *Reflux* is particular because its value is not always available for all the source problems. As explained before, the case base is available for two chemical engineering unit operations: distillation and absorption. But only distillation has a reflux ratio (indeed in absorption there is no reflux system). Consequently, the value of this attribute can direct the research of similar source cases towards distillation cases with values greater than 0 (in fact the value of the Reflux ratio) or towards absorption cases with a value equal to 0. However, it is an important operating parameter because it imposes the flow rate inside the column and consequently the purity of the outlet streams and the dimensions of the column.

2.3. Similarity measurement

The cases representation and the similarity measurement for case retrieval are strongly linked. The goal of the similarity measurement is to establish resembles and more precisely the degree of similarity between the target problem and source ones. The target problem (X) is compared with a source problem (Y) in the case base by the way of the global similarity measurement (1)

$$SIM(X,Y) = \frac{\sum_i w_i sim(x_i,y_i)}{\sum_i w_i} \quad (1)$$

Sometimes, for the problem description some attribute values are temporarily missing, because we need a time consuming additional analysis to reach them. This deeper analysis must be done later in the detailed design, but it is not always mandatory in the preliminary design. Indeed in this earlier design step, we have to reduce the number of potential solutions by making the best choice with imprecise or even missing values. Therefore we add the option IGNORE, when that option is activated for one feature, it is not taken into account for the global similarity measurement.

The global similarity criteria allow to rank all the source cases from the most similar to the less similar. The global similarity calculation is reached by the weighted (w_i) sum of local similarities: $sim(x_i,y_i)$. The former are used to express the different importance between features. The user can assign himself the weight values, or we can help him by asking him to classify the attributes according to their order of importance. Attributes with rank 1 are the most important, and two attributes can have the same rank. For each attribute, its corresponding weight is calculated by

$$w_i = 1 - \frac{rank_i - 1}{Max(rank_i)} \quad (2)$$

The local similarities are used to compute similarities between values of single attributes. They are calculated for each attribute (i) by comparison of the value of the target problem (x_i) with the corresponding source problem one (y_i). However, as problem features are described by different types of values (nominal or numeric), the local similarity calculation depends on these types.

For our problem description, the attribute mixture which describes all the chemical compounds is a nominal attribute specific to the application domain. Most of the time, for the features with nominal values, the following local similarity measurement is used:

$$sim(x_i,y_i) = \begin{cases} 1 \text{ if } x_i = y_i \\ 0 \text{ if } x_i \neq y_i \end{cases} \quad (3)$$

But in the case of chemical compounds, some chemical ontologies can be used in order to calculate more precisely this local similarity. In their CBR system, (Lieber and Napoli, 1996) represent chemical compounds by a frame with two attributes: atoms and chemical bonds. This representation is very useful for the purpose of their CBR system, i.e. to build a target molecule from simple products (the target molecule is given and the whole synthesis plan is building by the system), but here it would be inappropriate. Always with the idea to have a deeper similarity measurement than Eq. (3), another chemical ontology is used (Avramenko and Kraslawski, 2005). For each case, the feature "mixture" is composed of all the chemical compounds which will be separated in the distillation column. Obviously, the compounds are different from one case to another, but unfortunately the number of compounds encompassed in the mixture is also variant. The local similarity between two mixtures is calculated with a two steps method. First, the similarity between two chemical compounds must be evaluated, based on their chemical structure. This similarity is called binary local similarity (bsim) (we used the word "binary" because a mixture composed of two compounds is called a binary in chemical engineering). The binary local similarity is calculated for every possible pairs of two compounds with one compound belonging to the target mixture and the other one to the source mixture.

To calculate this binary local similarity, all the chemical compounds are divided into classes, subclasses and a hierarchical structure tree is built to describe the relations between them, Fig. 3 sketches this hierarchical structure. The first level nodes in the tree correspond to a basic group (organic or non-organic compounds), the daughter nodes correspond to classes and subclasses like: hydrocarbons, acids. A numerical value is assigned to each tree nodes, then the binary local similarity value for two compounds depends on their first common node in the tree. The deeper the common node is, the higher the binary local similarity is. For example, for the same compounds the local similarity is 1, and 0.9 for two different compounds belonging to the same family like alcohol; for instance ethanol and methanol in Table 3. For water and ethanol the first common nodes is organic with $bsim = 0.1$. The first step of the method ends with the building of the binary local similarity matrix: all the compounds of the target problem are located in the rows and in the columns

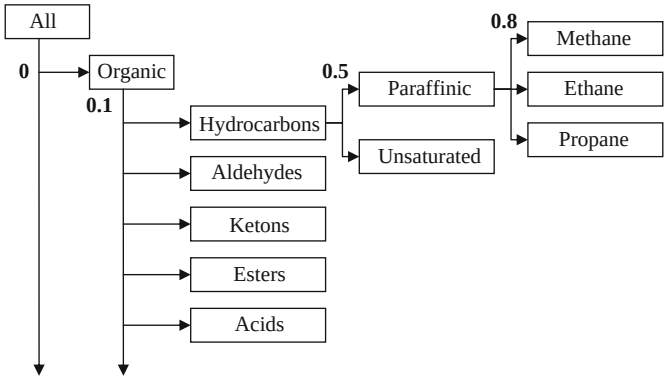

Fig. 3. Similarity tree for chemical compounds (Avramenko et al., 2004).

Table 3
Example of local similarity calculation for the feature mixture.

Target	Source			
	Ethanol	Water	Acetic acid	Ethyl acetate
Methanol	$bsim_{11}=0.9$	0.1	(0.1)	0.1
Ethanol	(1)	0.1	0.1	0.1
Water	0.1	(1)	0.1	0.1

for the source ones. On the crossing between a row and a column there is the value of the binary local similarity between this two compounds, Table 3. The goal of this matrix is to generate every possible pairs which will be needed in the second step.

This second step consists in finding and selecting the most similar pairs of components by maximizing Eq. (4) under the constraint that if a compound is embedded in a pair it cannot be in another one:

$$sim(x_m,y_m)=\frac{1}{m}\sum_{i=1}^{nt}\sum_{j=1}^{ns}x_{ij}bsim_{ij} \tag{4}$$

With $bsim_{ij}$ the value of the binary local similarity between compounds i and j, and x_{ij} a Boolean; $x_{ij}=1$ if the binary local similarity between compound i and j is chosen, else $x_{ij}=0$ (x_{ij} is equivalent to an affectation variable). $m=max(nt, ns)$, because the number of compounds in the target mixture (nt) can be different from the source case one (ns), Table 3. Finally, the local similarity for the feature mixture is calculated by solving system (5)

$$\underset{xij}{Max}\left(\frac{1}{m}\sum_{i=1}^{nt}\sum_{j=1}^{ns}x_{ij}bsim_{ij}\right) \tag{5}$$

With

$$\forall j\sum_{i=1}^{nt}x_{ij}\le 1 \text{ and } \forall i\sum_{j\le=1}^{ns}x_{ij}=1$$

For the example in Table 3, $sim(x_m,y_m)=0.525$, and the retained binary similarities are encircled. In this table, the binary local similarity between methanol and acetic acid is retained, but it can be noticed that ethyl acetate could be chosen instead of acetic acid because both hold the same value (no influence on the local similarity value).

Concerning the local similarity for numerical features, it is often calculated with a distance measurement. Most of the time, this distance is normalized by the domain definition Int_i ($Int_i=$ maximum value − minimum value for the ith feature) in order to avoid a distorted result because of the different amplitudes of variation between features. For example the pressure can vary between 0.1 and 70 atm in our system and temperature between 273 and 726 K.

$$dist(x_i,y_i)=\frac{|x_i-y_i|}{Int_i} \tag{6}$$

and

$$sim(x_i,y_i)=1-\frac{|x_i-y_i|}{Int_i} \tag{7}$$

The previous formula is very useful when you exactly know the feature values. But the problem description must be soften to account for some imperfect and fuzzy knowledges. We gather these imperfections in the term imprecision. In the earlier design stage, this imprecision arises from the requirement to model knowledge with information not precisely known or containing inaccuracies. Even if they are domain experts, sometimes they do not have a deep knowledge of the problem faced. They know an interval of possible values, upper or lower bounds, tolerable differences between the target problem and the source cases attribute values to ensure that a case is still relevant for their purpose. The designer must return such nuances in the problem description, especially in the preliminary design stage, as Fig. 4 demonstrates (Giachetti et al., 1997). Of course, if in the problem description the imprecision is modelised by a set of possible values instead of only one value, the local similarity measurement is directly impacted. All these requirements are satisfied with the fuzzy set theory, (Zadeh, 1965). A fuzzy set S in a domain D is defined by a characteristic function μ_s, which has values in the range [0,1]. $\mu_s(z)$ indicates the degree to which z is a possible value in the subset S.

Without the fuzzy set theory, when he describes its problem, the designer fills the numerical attributes with one value (the central value). With the fuzzy set representation, he has to give additional data to describe the domain of the possible values. For each numerical feature, three informations are necessary to build the associated characteristic function:

- A central value (or two for the relation *between*): c_i (or c_1 and c_2 for *between*)
- An imprecision: percentage of variation around the central value: λ
- A relationship with respect to the central value: *inf*, *inf-equ*, *sup*, *sup-equ*, *equal*, or *between*

For the five first relations, a triangular shape is used for the domain representation and a trapezoidal one for the relation *between*, Fig. 5. With these distributions, we can create the characteristic function shapes for each attribute (μ_{si}), Tables 4 and 5. Finally, $\mu_{si}(y_i)$ is the value of the local similarity between the source and target attributes

$$sim(x_i,y_i)=sim(c_i,y_i)=\mu_{si}(y_i) \tag{8}$$

With "$\lambda=0$" and the relationship "*equal*", we find the local similarity calculated with Eq. 7. To have more details on this part, Negny and Le Lann (2008) present the local similarity measurement with the fuzzy set theory.

However an effective retrieval is to find useful source cases to propose a solution. In this context, the choice of a retrieved case only based on similarity measures often reaches its limit. Indeed, in its current form, the similarity measure is not appropriated to estimate the relevance of a case for a given initial problem. In addition to the metric distance, the similarity measure must also

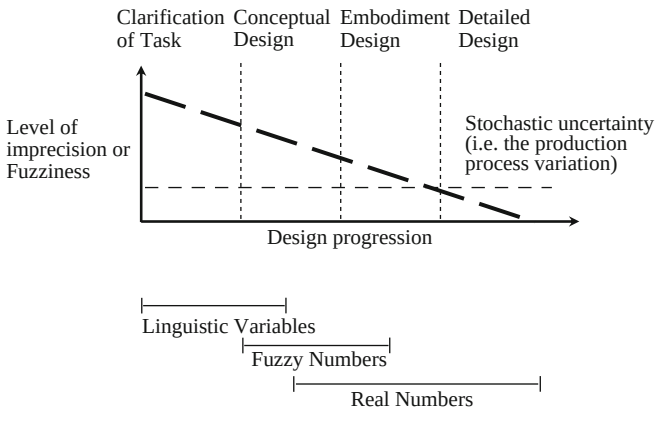

Fig. 4. Design stages versus imprecision and type of variables (Giachetti et al., 1997).

evaluate if a case is relevant with respect to adaptation, i.e. if it is easily adaptable. This view is detailed in the next part.

2.4. Retrieval guided by adaptation

In artificial intelligence, one of the main assumption usually stated is that similar experiences can guide future reasoning and problem solving (similarity assumption). The traditional similarity measures are a central and crucial stage in the CBR cycle because they strongly influence the entire problem solving process. But the success and efficacy of a CBR depends on the retrieval of a relevant case to solve the target problem; i.e. a case that can be easily and successfully reused to propose a suitable target solution. Consequently, the most similar case is not necessary the most appropriate for the adaptation purpose.

The adaptation process will be more or less expensive, depending on quality and utility of the retrieved case. Consequently, any attempt to facilitate adaptation is judicious for CBR efficacy. We cannot pretend to adapt easily cases if it is not anticipated in the retrieve step.

To achieve this goal, we are confronted to knowledge acquisition in order to model the required similarity measures.

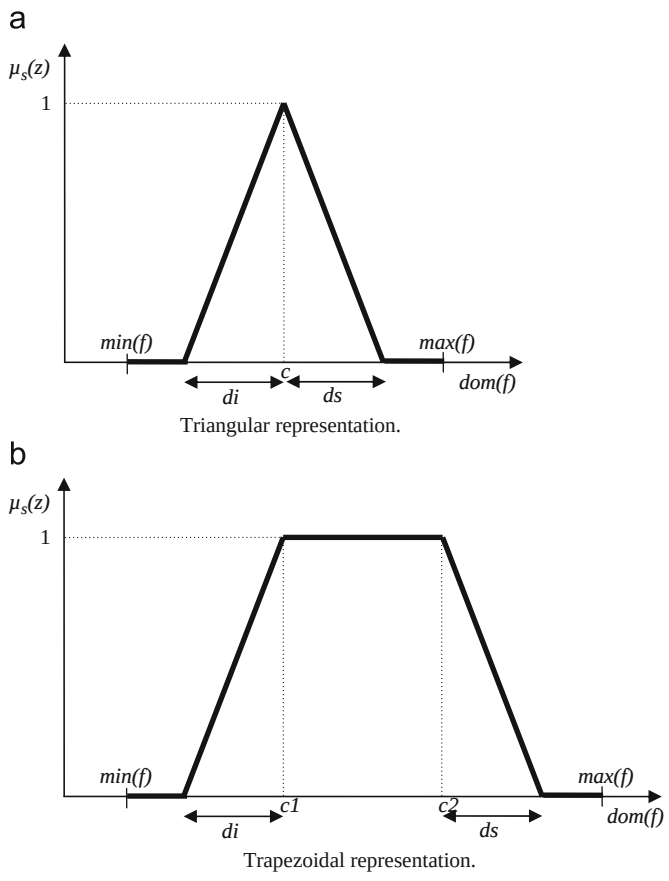

Fig. 5. a: Triangular representation. b: Trapezoidal representation.

The approximation of the utility of a case can be reached by including new domain knowledge. The mere and most used form of utility approximation are features weights. But weights represent a small part of the needed domain knowledge. To guarantee an efficient retrieval, additional knowledge must be acquired and formalized. Acquiring and modelling this similarity measures knowledge is a complex and time consuming task. Stahl and Gabel (2003) and Stahl (2004) have proposed a new learning approach which is based on the feedback about cases actual utility. This approach is composed with two different learning algorithms for optimizing feature weights in one hand, and the local similarity measures in the other hand (Stahl and Gabel, 2006). In another method Smyth and Kean (1998) have proposed to improve similarity measure by introducing adaptability criteria: research of cases more easily adaptable. Their idea was that similarity on superficial features needs to be increased by deeper knowledge about the significance of this features. They have called their technique "adaptation guided retrieval". The advantages of this technique are the selection of the most adaptable case and the link between similarity and adaptation requirements. Leake et al. (1997) had proposed another approach to link both of them. For each relevant case, an adaptation cost was evaluated based on knowledge inside the domain of application. Mille and Herbaux (2007) and Aarts and Rousu (1996) have implemented this idea with specific cost functions. The approach of Lieber (1999) and Lieber et al. (2001) was based on path similarity. Whatever the approach, CBR systems with a link between the retrieve and reuse steps give results that reduce adaptation failure.

Pralus and Geneste (2007) have developed a generic method to evaluate adaptability of a source case, this method is implemented in our CBR system. Consequently, the selection of a source case is based on two criteria: similarity and adaptability.

In the first phase of the method implemented, an adaptation space is built for the source solution and then the adaptability is calculated. The easiness of adaptation of a case is directly linked to the potential of solution values that its adaptation space contains. The target solution will be researched in its adaptation space. Like for similarity measurement, the global adaptability of a source case corresponds to an aggregation of the different local adaptability of each features (9)

$$ad_s = \sum_{i=1}^{n} ad_i/n \text{ (if necessary a weighted sum can be used too)} \tag{9}$$

To build the adaptation domain for one attribute of the target solution, the fuzzy sets are also used. When a source case is stored

Table 5
Parameter values for the characteristic function in trapezoidal representation.

	Between
c1 =	*v1*
c2 =	*v2*
di =	$min(\lambda v1;\ v - \text{-}min(f))$
ds =	$min(\lambda v2;\ max(f) - v2)$

Table 4
Parameter values for the characteristic function in triangular representation.

	Equ	***sup, sup-equ***	***inf, inf-equ***
C =	*v*	*max(f)*	*min(f)*
di =	$min(\lambda v;\ v - min(f))$	$min(max(f) - (v - \lambda v);\ max(f) - min(f))$	*0*
ds =	$min(\lambda v;\ max(f) - v)$	*0*	$min((v + \lambda v)\text{-}min(f);\ max(f) - min(f))$

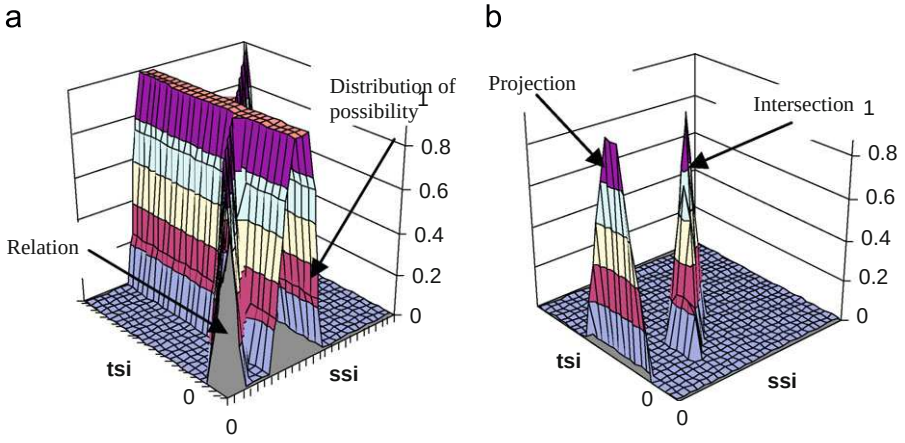

Fig. 6. a and b: Graphical representation of the adaptation domain building.

in the case base, the domain expert does not characterise a source solution attribute with one and only one value but with a distribution of possibility around this value, ssi axis in Fig. 6a. Moreover, in the reuse step, the domain expert expresses its opinion on how the values of the source solution features can be exploited to find a solution. Here again, the fuzzy set theory is used. Like for each problem feature in similarity measurement, for each solution feature i the expert specifies a relation and a percentage of imprecision, i.e. definition of μ_{soli}. For one solution feature i, the relation μ_{soli} allows to situate the searched value for the target solution feature (tsi) with respect to the values of the corresponding source solution feature (ssi). In Fig. 6a, we represent the similarity function equal $\pm\lambda\%$. This expresses that for the solution feature i, tsi must be equal to ssi with an imprecision around $\pm\lambda\%$.

The projection on the tsi axis of the intersection between the similarity function (μ_{soli}) and the distribution of possibility for ssi, gives the distribution of possibility for tsi. The graphical representations of these intersection and projection are illustrated in Fig. 6b.

The form of the shape created after projection determines the set of possible values for tsi and it traduces the easiness of adaptation for the selected source case. Indeed, the shape of a fuzzy set traduces its fuzzy level. The more the form is wide, i.e. fuzzy or imprecise, the more it contains values and the highest the local adaptability of this feature is. The specificity of a fuzzy set (Yager, 1992) allows to measure the degree to which a fuzzy set contains one and only one element:

$$S_p(F) = \int_0^1 \frac{1}{\sup F_\alpha - \inf F_\alpha} d\alpha \tag{10}$$

Sup F_α (resp. Inf F_α) represents the upper (resp. lower) bound of an α cut on the domain. The specificity value is in the range [0;1], with $S_p=1$ for a set with one and only one possible value. As assumed before, the less specific a set is, the more adaptable the attributes is. Therefore, the local adaptability can be calculated with the following formula:

$$ad_i = 1 - S_p(F) \tag{11}$$

The global adaptability of a source solution is calculated with all the local ones.

This method uses additional expert knowledge to find adaptable case. Here domain expert point of view is mandatory to select the right fuzzy set (μ_{soli}). But for none expert, the problem of finding an adaptable case is shifted to find the right fuzzy set. More generally, as they need additional knowledge, most of adaptation guided retrieval techniques need to be configured with expert experiences. Consequently, they move the adaptability problem towards a configuration problem which needs less expert knowledge. Nevertheless, they are useful in order to anticipate the retrieval by filtering cases with a low potential of adaptation or worse impossible to adapt.

3. Retrieval in CBR

3.1. Architecture

The case base is the central part of any CBR system (Fig. 1), since it represents the experiences to be used by the system in order to solve new problems. Two modules are directly connected to our case base in order to take advantage of the stored knowledge. Each module gathers some mechanisms corresponding to the different steps of the CBR, Fig. 7.

The development of a CBR system, even simple, involves a number of steps such as: modelling a suitable case representation, defining a similarity measure, implementing a retrieval method and maintaining the system. Our software is composed with two different modules one for case base administration and the other one for case exploitation.

The first module is dedicated to the domain expert in order to define and configure its system. This task needs to be done by a collaboration between a knowledge engineer and a domain expert, because valuable knowledge must be appropriately coded by a knowledge engineer (who is aware about the technical aspects of all the CBR steps). This includes:

- The case representation: Currently, the choice is made between the relational database technique and the object oriented approach.
- The similarity measurement: Specification of appropriated way to calculate local similarity for each features with the possibility to choose between several methods to customize formula (1), and also to choose the technique to fill the weights.
- The case indexing method with its specific parameterization (discretization steps for the query sphere, the decision sequence and the bounds on the decision variables for the indexing trees, detailed in the next section). When a new case base indexation is created, it can be stored in the index base.
- Maintenance: This part displays all the source cases. They can be modified, expanded or removed, and obviously the expert can create new ones.

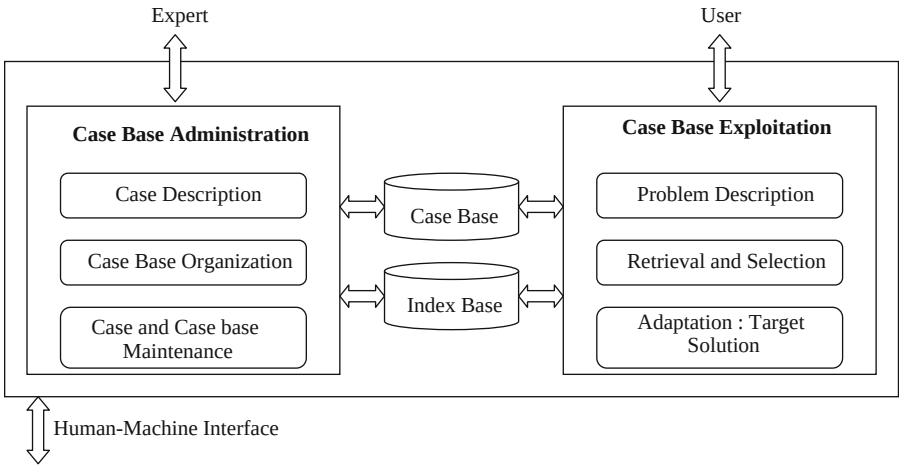

Fig. 7. Case base architecture.

The user is connected to the second module via a specific human machine interface. He defines its current target problem, fills all the features, eventually selects the local similarity formula, customizes the global similarity, defines the way to calculate weights, selects the indexing methods…. After this step, the retrieval of past experiences is started, and he displays all the retrieved cases. The latter are sorted with respect to both criteria; global similarity or adaptability. The last step deals with the adaptation of the selected source case(s) for proposing a target solution.

3.2. Case base organization

Thanks to the retain steps of the CBR approach, the case base grows up quickly, therefore it needs to be structured to improve the retrieval of relevant cases when queried. When creating a case base, it is necessary to consider the following aspects (Pal and Shiu, 2004):

- The structure and representations of cases.
- The memory model retained for organizing the whole case base.
- The indexation used to identify each case.

Numerous approaches can be applied to index cases for efficient retrieval. A flat case base is the more simple and common organization. Here, all the cases are stored at the same hierarchical level, i.e. the root node, and the query case is compared with each case in the memory. The k nearest neighbours algorithm is commonly used in CBR systems to retrieve relevant cases because of its simplicity and robustness. During the last years, this algorithm was improved: attribute values have been weighted, cases themselves have been weighted (Anand et al., 1998). Despite these improvements, the major drawback of the flat case base remains an exhaustive search through the whole memory. Unfortunately, this drawback increases sharply when both the number of features or the number of cases in the base becomes important: tremendous computational effort. Some methods were developed to avoid an exhaustive search, for instance the use of genetic algorithms.

Another solution for reducing the search processing time is to index the memory by proposing a case indexation where cases are gathered into categories to reduce the number of available cases for similarity measurement. Case indexation refers to portioning the memory for a faster reliable extraction of relevant subset of cases. The choice of the case base organization is important to allow an efficient retrieval. As Pal and Shiu (2004) explained, the index should be abstract enough to enable retrieval in all circumstances, but not too abstract because the case may be retrieved in too many situations and it could lead to an important computational efforts to match cases. Several researchers have developed very specific indexing methods for their CBR applications. Most of these methods are too specific and cannot be extended to other domains (Deangdej et al., 1996; Fox and Leake, 1995). In the literature there are several other indexing methods for CBR and large databases like the Bayesian model (Pal and Shiu, 2004), a prototype based neural network (Malek, 1995, 2000), genetic algorithms (Bueno et al., 2007) and the k-medoid based algorithm (Barioni et al., 2008). More generic indexing methods coming from machine learning and data mining communities have also been successfully applied in CBR systems.

With flat case base, another common memory organization is hierarchical structures. In CBR, the more famous hierarchical structures and the more widely applied are the indexation trees with their various improvements and modifications. Trees result in recursively portioning a data set into subsets at each nodes. The nodes at the bottom of the tree are called leafs and these above are inner nodes. Inner nodes contain values or interval values to separate cases, and at the leaf nodes we have the information on the cases locations. Finally, leaves represent the classification and branches the conjunction that leads to this classification, Fig. 8. The typical trees used in CBR are B-trees, B^+-trees, B*-trees and the improved R-trees, R^+-trees, R*-trees which take into account range and multidimensional searching. The INRECA tree is probably the most successful decision tree, developed in the INRECA and INRECA-II projects. Besides, with the development of database management systems, these hierarchical trees are continuously ameliorated by different researchers: X-tree by Berchtold et al. (1996), TV-tree by Lin et al. (1994), M-tree by Ciaccia et al. (1997), DBM tree by Vieira et al. (2004) and Slim-tree by Traina et al. (2000). A review on some of these metric trees is presented in Hjaltason and Samet (2003). For example, in the Slim-trees, the elements are gathered into disks of fixed size, each one corresponding to a tree node. Like traditional trees, the elements are stored in the leaves. But in this method, each node has one element considered as representative for all elements stored at that node, and a covering radius, Fig. 8.

The metric trees are built to search elements within large sets of data. These approaches are not yet implemented in CBR

systems, but they open possibilities for case representation (images, video, audio, DNA sequences) and they give very interesting ways to explore for case base indexation.

Nevertheless, from a CBR point of view they have the same major drawback as the traditional B-tree, R-tree and their further evolutions. They are also based on an exact matching or well-defined boundary. As a consequence, the cases are within one and only one range; overlapping is not allowed. Let present a mere example: assume that the target case has an index value near a boundary, i.e. $p=14$ in Fig. 9. Although if there is a record with an index just next to it: $p=15$ in Fig. 9, it will not be retrieved if it is in another range than the target one. In Fig. 9, with a target problem with $p=14$, the search is focused in the range $1 \leq p < 15$, therefore cases with $p=15$ will not be considered during retrieval. In some metric trees, the overlapping is possible but it is due to a very large database, there are many suggestions to reduce it because the degree of overlap directly affects the performances of algorithms during retrieval.

Galushka and Patterson (2006) and Patterson et al. (2002a, 2002b) have developed another generic index called discretized highest similarity D-HS with its various versions. The main goal of this generic approach is to create an efficient and domain independent indexing structure available for a wide range of CBR systems. All dimensions for mapping case attributes are split into intervals. For nominal features, the number of intervals equals the number of discrete values, for numerical ones, this number is predefined for the D-HS^M version of the algorithm. The indexing process consists in locating cases into the subspace delimited by each of their attributes values. During the retrieval stage, the process identifies the corresponding subspace of the query case, and extracts all the cases included in this subspace. An example of the indexing method is illustrated on Fig. 10 for a mere graphical representation in only two dimensions; x and y represent two numerical features. Both attributes are discretized into 4 intervals, respectively, labelled C_1^x, C_2^x, C_3^x, C_4^x for x and C_1^y, C_2^y,

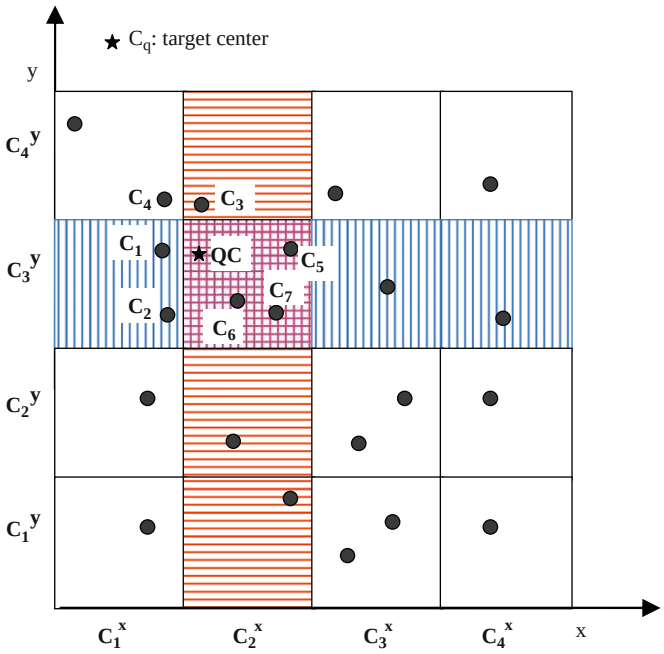

Fig. 10. D-HS in two dimensions.

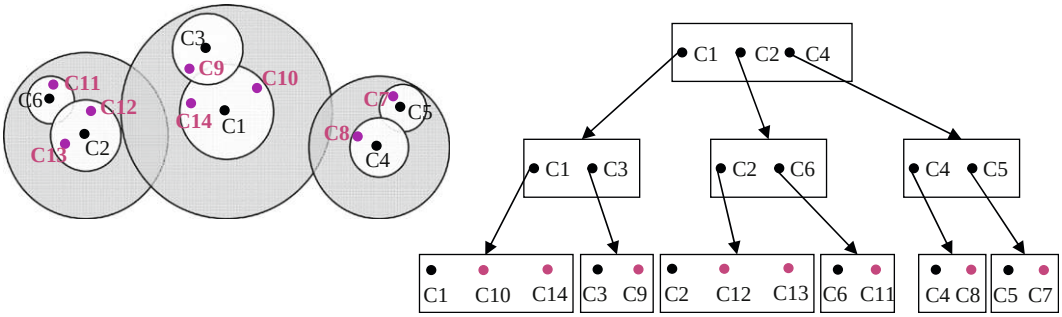

Fig. 8. Slim tree representation.

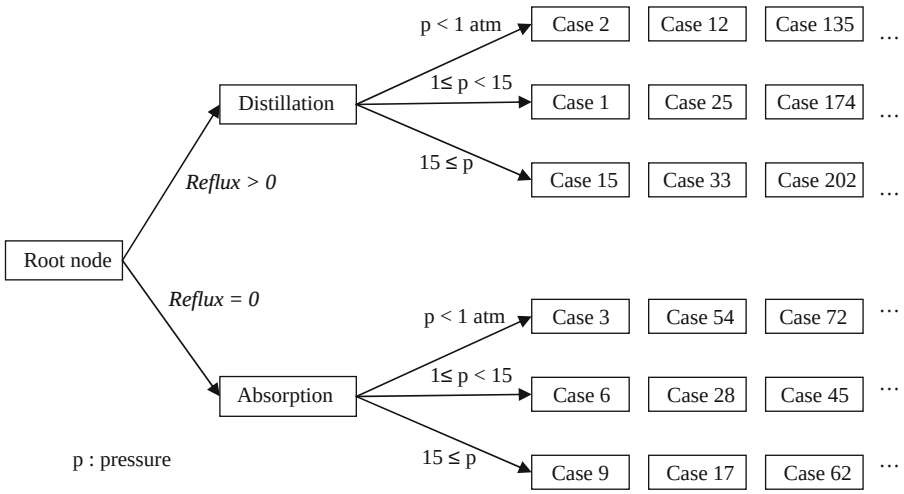

Fig. 9. Example of a decision tree index.

C_3^y, C_4^y for y. Let us assume that thanks to its attribute values, respectively, on x and y, the query case QC (i.e. target problem), falls in the interval with index 2 for x and 3 for y. The crossing between C_2^x and C_3^y isolates the subspace with the square bottom, and the final retrieval set of cases FC=$\{C_5, C_6, C_7\}$. The similarity measurement is achieved only for the cases encompassed in FC in order to rank them.

This indexing method is very efficient especially in large case base. It needs low knowledge engineering overheads and it is easy of maintenance. However it presents two major weaknesses. First, with the discretization process, it is impossible to predict in advance the local density of cases isolated in the final subset FC. In the worst case, it can gather no case or in the opposite a huge number of cases, which would affect the efficiency of the D-HS^M algorithm. This problem was solved with the D-HS^E version that uses entropy to find optimal discretizations. The second inconvenient, persistent in the new version, appears when QC is near a boundary. In practice, all the cases included in the subspaces surrounding the query case are not taken into account but unfortunately they could be relevant. This is especially true when QC is near a corner or a boundary, the neighbouring cases outside the subspace are not considered, cases C_1, C_2, C_3 and C_4 in Fig. 10. In the next section we address this boundary case issue with our algorithm.

3.3. Query sphere algorithm

In this part we propose another generic algorithm to search relevant cases in a descritized case base avoiding the boundary case issue. The neighbourhood query problem consists in finding the relevant cases within a given distance from a given center location QC, i.e. target problem. For that purpose we adapt the spherical indexing method presented by Brodu (in press) for creating an efficient domain independent indexing method.

Let us define an area around the target case which gathers a number of nearby cases. These cases are considered closest, according to the similarity measure, because they are stored in the vicinity of the query case (QC). We can suppose that similar cases are stored in the same subspace of the case base. Let B represent the entire case base, and SC^j a source case j represented by a vector of attributes; SC_i^j is the feature i of the source case j (with $j=1$ to N_c and $i=1$ to N_a).

Besides, let us assume that all the case attributes are numeric, we discuss the problem of nominal ones at the end of this part. For a QC, retrieving the final subset (FS) of relevant cases is equivalent to find the cases in an area near the location of QC. For each dimension i, we can define a width d_i to delimit an interval of acceptable values around the attribute value I of the target problem QC_i: $[QC_i-d_i;\ QC_i+d_i]$. Only the cases with $SC_i \in [QC_i-d_i;\ QC_i+d_i]$ are considered. The intersection of all these intervals along each dimension determines the relevant area close to QC, then FS can be defined

$$FS = \{SC^j / \forall i = 1 \ldots N_a\ SC_i^j \in [QC_i - d_i;\ QC_i + d_i]\} \tag{12}$$

The strength of this technique is that QC is in the center of the search area. Unfortunately, a computational costly exhaustive search has to be carried out in order to define which SC^j belongs to the relevant area. To make the retrieval process more efficient, avoiding the exhaustive search, a grid structure as in Galushka and Patterson (2006) is proposed. For each attribute, the range of the possible values is divided by a predefined number of sub-intervals. This leads to the discretization of B along all its dimensions. Like in the D-HS method, the indexing process consists in locating each SC^j in its corresponding cell. Each SC^j is located by its coordinates in the case base grid SC_i^j. Thus each SC^j is affected to one and only one cell. With this spacial discretization all cells that are beyond a search distance (d_s) are automatically and quickly eliminated therefore all the cases within these cells are also eliminated without calculating their distance to QC. Only cells with a distance below d_s are considered. These cells delimit a hypercube around QC (a cube in 3 dimensions).

With this algorithm the boundary problem is avoided because QC is located on the center of the research area. But the number of cells to consider increases, resulting in a decrease of the retrieval process efficiency. To address this problem, instead of considering the hypercube around the center cell (CC, cell containing QC), we use an hypersphere (a sphere in 3 dimensions). Indeed, the volume of the hypersphere is sharply lower than its bounding hypercube. The number of cells strictly included inside or crossed by a hypervolume is directly correlated to the volume, thus the hypersphere contains less cells than the hypercube. In 3 dimensions, if d_s is the search distance (i.e. radius for the sphere and half the length for the cube), the volume ratio between the sphere and the cube is $Vs/Vc=\pi/6=52\%$, so the sphere fills about 52% of the cube. This ratio represents the percentage of common cells between the sphere and the cube. More precisely, it is the lower limit reached only when the discretization lengths for B tend to zero. For higher dimensions this ratio decreases: 31% for 4 dimensions, and around 8% for 6 dimensions. The idea of the query sphere indexing is to consider the hypersphere instead the hypercube for the research which decreases drastically the number of cells to explore by eliminating cells inside the hypercube that do not intersect the hypersphere. The computational cost to set up the spherical indexing is lower than the cost of considering all the cells inside the hypercube. But the goal of the first step of the query sphere algorithm is to extract the common cells between both hypervolumes.

Whatever a source case j in B, $d_t^j = \|SC^j - QC\|$ defines its true distance from QC. The final purpose of the query sphere algorithm is to find and to extract the subset of cases satisfying

$$d_t^j \le d_s \tag{13}$$

In a first phase, the reasoning is driven on cells and not on cases. To consider a cell or not, the distance between the center cell and the other cells in the base is evaluated, its corresponds to the minimum distance d_c between two cases inside both of them. For one arbitrary reference cell, this minimum squared distance can be pre-computed (only once and before to start the retrieval step) and stored in a table, Fig. 11a represents this pattern for 2 dimensions problem. Once the QC defined, the CC is identified, then the previous pattern is centred on CC. Then for a given search distance d_s, d_s^2 is flanked by two successive integers: $n \le d_s^2 < n+1$. Therefore only the cells with a label number smaller or equal to d_s^2 are kept. For example, for $d_s^2=4.1$ all the cells with a label number greater than 4 are automatically excluded. In Fig. 11b, the

a

18	13	10	9	9	9	10	13	18
13	8	5	4	4	4	5	8	13
10	5	2	1	1	1	2	5	10
9	4	1	0	0	0	1	4	9
9	4	1	0	0	0	1	4	9
9	4	1	0	0	0	1	4	9
10	5	2	1	1	1	2	5	10
13	8	5	4	4	4	5	8	13
18	13	10	9	9	9	10	13	18

b

18	13	10	9	9	9	10	13	18
13	8	5	4	4	4	5	8	13
10	5	2	1	1	1	2	5	10
9	4	1	0	0	0	1	4	9
9	4	1	0	QC	0	1	4	9
9	4	1	0	0	0	1	4	9
10	5	2	1	1	1	2	5	10
13	8	5	4	4	4	5	8	13
18	13	10	9	9	9	10	13	18

Fig. 11. Minimum squared distance between cells d_c^2

retained cells are those inside the bold perimeter. In the algorithm we also take into account the particular case when for a given CC and radius d_s the pattern can map cells outside the memory. In this particular situation, only cells inside the memory are considered thanks to a binary linear index. In this first step we generate the first version of the subset of retained cells (Cellsubset1).

Until now, only the cells containing CC was considered. But as Fig. 11b illustrates, some additional cells can be eliminated from this first subset. Always with $d_s^2=4.1$, the cells with the bold label number 4, outside the grey circle must be rejected. Indeed, while they are not d_s away from CC they are d_s away from QC. To overcome this issue, the location of QC inside the center cell must be handled. Consequently another test dealing with QC location in its cell and the minimum distance between QC and the other cells is implemented. In its cell, QC is located by its distances ξ_k to each boundary of the cell, Fig. 12 for 2 dimensions. The mathematical assumptions and demonstration about this test are detailed in Brodu (in press) and in Appendix A. The final conclusion is that in each direction k (each dimension has two directions) there is the possibility that

$$\{d_s\} < \xi_k \text{ or not} \tag{14}$$

With $\{d_s\}$: fractional part of the distance.

To be general this test must take into account all the possible position of QC inside its cell, this gives 2^{2Na} combinations ($2^6=64$ combinations in $N_a=3$ dimensions) leading to cells to reject. As the test (14) is based on a Boolean value in each direction, these different 2^{2Na} combinations corresponding to each possible QC position within its cell, can be pre-computed resulting in as many tables containing the cells to retain.

Unfortunately, Eq. (14) is only tested in run time because we need the QC location. In fact, once QC location known, ξ_k is calculated in each direction and a Boolean vector with the results of test (14) in each direction is generated. Then the right table corresponding to the current Boolean vector is selected. Nevertheless, this additional test does not concern all the cells in Cellsubset1, but the table should be used only for $d_c \geq \lfloor d_s \rfloor$ ($\lfloor d_s \rfloor = \text{floor}(d_s)$ is the largest integer below or equal to d_s). Indeed, the other cells satisfy obligatorily this second test. In the algorithm, the pattern is used from the center cell to the others by increasing the distance. When the distance $\lfloor d_s \rfloor$ is reached, the selection of the correct table is activated. This leads to the shorter Cellsubset2.

In one hand, cells excluded by this second test are really to reject. On the other hand, this second test does not strictly meet its target. Indeed, because of an assumption during the mathematical demonstration, some cells satisfying this test are in reality d_s away from QC (Appendix A). Unfortunately, at this step of the algorithm, these cells escape from rejection for example the encircled 2 in Fig. 11b is still in Cellsubset2. This cell could be rejected at a cost of an additional check, i.e. third test.

The real distance d_t is used for this test: cells with $d_t^2 > d_s^2$ are rejected. Here again, not all the cells of Cellsubset2 are tested but only some of them present so far; cells above d_s-1, heuristic proposed by Brodu (in press).

After the whole rejection algorithm, the Cellsubset3 of the cells to consider for similarity and adaptability measurements is established. But generally, case base has attributes values unevenly distributed, therefore some cells could contain no case. In our approach the empty cells are labelled and then automatically removed from Cellsubset3. Finally, the global similarity and adaptability are calculated for all the cases included in cells belonging to the final subset (FinalCellsubset). Fig. 13 summarizes the query sphere algorithm, but in reality step 1 and step 2 are coupled because test (14) is activated during the creation of Cellsubset1.

The whole rejection algorithm is very attractive because the processing overhead is reduced. Indeed, the processing time cost to reject cells is lower than measuring similarity and adaptability for all the cases encompassed in the hypercube, because some parts are pre-computed. Moreover, with this process we reach all the cases stored in the neighbourhood of QC, the boundary case problem is removed.

The moment to calculate the adaptability criteria can be done with two different strategies. First, the similarity and adaptability are calculated for all the cases in the base (flat case base). But when the case base contains numerous cases the processing time to estimate both criteria would become prohibitive. In a second strategy, some cases are extracted from the memory based on the similarity measure, then the adaptability criteria is evaluated in this subset of extracted cases. Here our strategy is a trade off between both of them. With our algorithm both criteria are calculated for all the cases in the final subset of cases. As we search cases in the neighbourhood of QC, and as we suppose that similar case are stored in this neighbourhood, our algorithm is

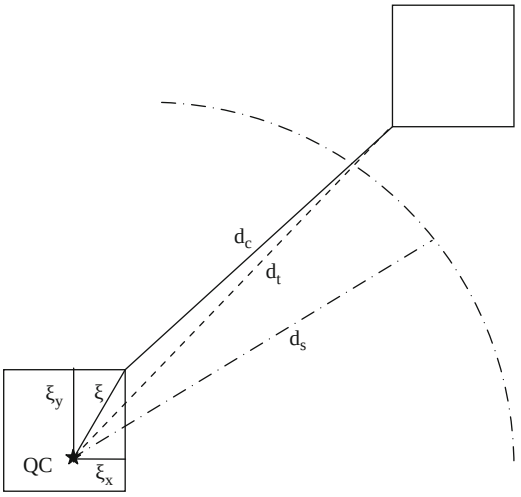

Fig. 12. QC location in its cell (in 2 dimensions).

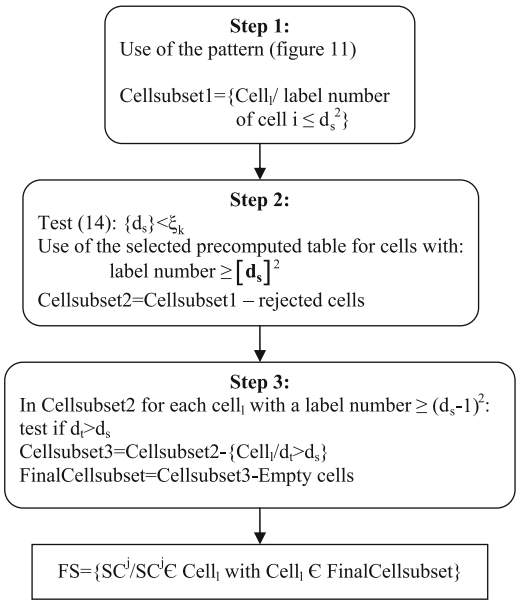

Fig. 13. Query sphere algorithm for selecting relevant cases.

equivalent to extract a subset of most similar cases and then to calculate adaptability.

Now we discuss the following assumption: *all the attributes are numeric*. Indeed, the algorithm needs a discretization along each attribute which is difficult for nominal value. At first sight, the number of intervals could be equal to the number of discrete attributes values. This method is implemented and gives good results when the number of discrete values is very small (less than 5) or when we can order them. As in the query sphere algorithm we need to define a search radius in each dimension, the order of the discrete values in the definition domain is crucial for retrieval. For example let us assume that we have a nominal attribute with a fixed discrete value and a search radius equal to one, only the discrete values situated before and after the current one are considered during retrieval. This leads to a very problematic situation when the domain of discrete value is wide because it is often difficult even impossible to rank them in a relevant way. On the other hand, taking into account all the discrete values are not a realistic solution because it leads to consider a significant number of cells, and therefore of cases, reducing the algorithm efficiency. This way to proceed is not restrictive enough. Besides, this issue is increased when the target problem is described with several nominal attributes. Consequently, the issue of nominal attributes is still one possible way of improvement of our algorithm.

4. Case study

4.1. Example

In this section, various options of our CBR system are tested through the distillation of a three components mixture; methanol/ethanol/water. Mori et al. (2006) have experimentally studied this distillation for different operating conditions: pressure, flow rates. These operating conditions define our target problem. The goal of this example is to compare the solution proposed by the CBR system with respect to the type of the column packing used by Mori et al. (2006).

In the target problem description, the option IGNORE is activated for the feature temperature. Indeed, the authors do not specify the temperature range of their experimental studies, consequently it is assumed unknown. Obviously for chemical engineering expert, this temperature range can be easily calculated by an additional thermodynamic analysis. But to exemplify how a partial problem description is treated, this thermodynamic analysis is not driven. After the target problem description, i.e. the filling of the problem attribute with the fuzzy sets, all the μ_{si} functions for the similarity measurement are automatically built in the CBR system.

The subset of the most relevant cases is extracted with both the decision tree index and the query sphere algorithm. These two algorithms are compared in the next part. The first retrieval is driven with the query sphere algorithm. Let us pay attention to the feature *pressure*, in order to put in highlight some weaknesses of the decision tree. Here, for our target problem the pressure is equal to 1 atm. The query sphere algorithm extracts and proposes a subset of the 20 most similar cases with pressure values both below and above 1 atm. More precisely, the majority of source problems have a pressure value below 1 atm, including the five most similar ones.

The decision tree restricts the case base with the following succession of feature evaluation: reflux at the root node, then pressure and finally on the inlet flow rate. In the decision sequence, the temperature is ignored because the option IGNORE is activated. Indeed, for a partial description of the target problem, the feature ignored cannot be used to discriminate the relevant cases. But in the decision tree drawn in Fig. 9, for the feature pressure, the upper and lower bounds values for the cases discrimination are not appropriated. Remembering that for our target problem *Pressure* = 1 atm, thus all the source cases with a value around 1 are relevant for this feature. But with the bounds values in Fig. 9, only source cases with *Pressure* ≥ 1 would be considered in the retrieval step. Unfortunately, for the target problem faced, the most relevant source cases have pressure below 1 atm, as found with the query sphere algorithm. In fact in distillation, and more generally in chemical engineering, the atmospheric pressure is a hinge value because under this one or for very high pressures we often need specific technological apparatus. Consequently this hinge value must be inside a range of values but not at a bound. Here we put in highlight one of the drawback of the decision tree indexation: it is mandatory to have expert knowledge to build a reliable and robust decision tree with an efficient decision sequence. Unlike the decision tree, the query sphere algorithm does not need additional knowledge for indexing the case base. Thanks to its higher level of abstraction and its domain independency, this indexing approach can be easily used by any type of users even none expert people.

Once the correct decision tree index established (in fact the decision tree in Fig. 9 is just used as an example to bring out the previous drawback, but it was not really constructed in our system), the two algorithms give the same list and ranking of the most relevant source cases. On the similarity criteria alone, our system ranks first a source case with a random packing (case 1) as solution ($SIM_{case1} = 0.83$), followed by two solutions with the same structured packing; cases 2 and 3 (respectively with $SIM_{case2} = 0.81$, $SIM_{case3} = 0.79$). It is important to notice that there is a huge technological gap between the two kinds of packing; random or structured ones (shown in Fig. 2). This important technological difference imposes that when the retrieved case gives one kind of packing, the adapted solution is necessarily in the same category of packing. For example, after adaptation it is impossible to propose a structured packing from a retrieved random one. Even with the introduction of additional adaptation knowledge, often needed in the adaptation phase, we cannot reach a kind of packing with a source solution offering the other one. Thus, if the selection is made only on the similarity criteria, the source solution 1 is retained (third column of Table 6). Unfortunately, this source solution is very difficult to adapt and gives a very remote solution from the real one, second column of Table 6. Indeed, the proposed target solution is a random packing in plastic. In some other examples, relying only on similarity, we extract source cases impossible to adapt, it depends on the operating conditions. Therefore, to avoid these adaptation difficulties, we have to anticipate them through the adaptability criteria.

When we add the adaptability criteria to the selection, the ranking of these previous source cases is different. The values of the adaptability criteria, respectively, are: $AD_{case1} = 0.75$, $AD_{case2} = 0.87$ and $AD_{case3} = 0.83$. With this additional information, the user wonders which case to select. But the adaptability measure clearly directs the choice towards cases 2 and 3, which propose a structured packing. Here we can notice that adaptability criteria are essential for the choice of a relevant case. Moreover, the user point of view is mandatory to deal with the two criteria for the selection.

When we consider both criteria, cases 2 and 3 are retained for the CBR next phase; adaptation. The two source solutions are composed of the same type of structured packing but they have different geometrical characteristics, specific areas and material. For this example we use the adaptation method presented by Avramenko et al. (2004). Finally, the proposed target solution is

Table 6
Packing solutions for the example.

	Proposed solution	(Mori et al., 2006) Solution	Most similar case C1
Type of packing	Structured Packing Montz pak B1 300	Structured Packing Montz pak B1 250	Random Packing Exlon Ring
Material	Stainless steel	Metal (not specified)	Plastic
Specific area (m^2/m^3)	350	247	220
Geometrical characteristics			**Geometrical characteristics**
Angle	45°	45°	Length (m): 0.05
Element height (m)	0.201	0.197	Free space: 90%
Corrugation height(m)	0.008	0.012	Bulk density (kg/m^3): 240
Corrugation base (m)	0.0167	0.0219	Number of pieces/m^3: 1250
Corrugation side length (m)	0.0116	0.016	
Packing illustration			

the Montz Pak B1 300 (first column in Table 6). This target solution is to compare with the real one. For this example, the CBR system gives a good preliminary solution for the resolution of the faced problem but some discrepancies still remain. The feature material is different. It is to notice that for the two selected source solutions, the material is also different; stainless steel and carbon steel, respectively, for case 2 and 3. The choice is directed towards stainless steel because the mixture of case 2 is most similar than the mixture of case 3 (under the same operating conditions magnitude). Here we stated the assumption that: the most similar the mixtures, the most reduced the risk of material degradation. This way to adapt is just a first approximation, obviously it needs to be improved because it is based on a specific rule to the domain studied which is not suitable. But as we explain in the introduction knowledge management with the rules based approach generates some difficulties. The future implementation of a more generic adaptation method is discussed in the conclusion.

4.2. Comparison of the two algorithms

Twenty-five examples were treated in the experiments in order to compare the query sphere indexing method with the decision tree one. In the comparison, the number of features included in the decision sequence is equal to the number of discretised features for the query sphere. Therefore, we are interested in assessing the general performance of both index methods over the 25 target problems. We assess the average performance along two criteria; by comparing the list of the relevant cases and by assessing their individual processing time to generate the list.

In all the experiments, the list and the rank of the most relevant source cases for these two methods are compared. This first comparison is made after similarity and adaptability calculation in order to have the rank of each relevant case. For 78% of the attempts, the list is exactly the same, i.e. same cases extracted and same rankings. Obviously, for one case the similarity and adaptability criteria have the same values with both methods because the measures depend only on the cases retrieved and not on the method to retrieve them. For the remaining 22%, the list is slightly different, but the majority of the discrepancies are for cases at the end of the list, relevant cases but the less similar and/or adaptable. Moreover for the remaining 22%, the different cases are those which have some features values near bounds of decision variables in the sequence. If bounds are changed we can suppose that we would find the same list with the two algorithms.

Concerning the processing time, the decision tree method outperforms the query sphere one for 60% of the attempts. It is important to underline that this results concerns only research time. The query sphere algorithm has better performances on this criteria when the research radius is small, or when there is a lot of source cases in the vicinity of the query center. In the latter case, the query sphere algorithm stops rapidly because it has found its k nearest-neighbours without the need to explore all the cells in the hypersphere. However, the difference on the two algorithms is slight on this criteria; the ratio between the processing times of both of them is $\pm 8\%$.

Besides, thanks to the retain steps the case base becomes wider and consequently the decision tree and the indexing must be updated in order to decrease the number of cases in the isolated subsets. With the query sphere algorithm, this issue is avoided because the case base is only re-indexed when the number of cases in the cells becomes too important. Therefore, the discretization needs to be refined, but it does not need to be re-indexed as frequently as in the decision tree.

5. Conclusion

In this work, we propose some ways to improve the retrieve step in case based reasoning. For the choice of the most appropriated and relevant source case(s), the traditional similarity measure is coupled with an additional deeper knowledge: adaptability. The fuzzy set theory is used to calculate this adaptability criteria. This second decision support criteria allows to anticipate the next step of the CBR cycle, adaptation. As it is shown in the example based on this criteria the expert can improve his decision and he avoids some future difficulties or failures in the reusing of the source solutions. Finally, the CBR system efficiency is improved.

In a second part, a new index method is presented in order to facilitate the retrieval of relevant cases. This new index method is based on the query sphere algorithm. The major strength of this algorithm is that it is generic, domain independent and it does not need some expert knowledge to have a reliable and efficient indexation of the case base. In terms of performance it is very close to the decision tree index. However, the growing of the case base has slight effects on this method. On the other side, with the increasing number of cases this method becomes more attractive with respect to the decision tree, because with the latter the case

base must be often re-indexed. As any method the query sphere algorithm has also some weaknesses:

- the discretization of the case base is only made on numerical features, because for nominal values it is more difficult.
- it demonstrates some difficulties in the situation whenever case attribute values were unevenly distributed.

These two points give the ways for future work to improve the algorithm. For the latter, an entropy based discretization approach can be used for each numerical attributes. This approach identifies good split points. Galushka and Patterson (2006) notice its effectiveness in their algorithm. The former is discussed at the end of part 3.3.

Another future development of this CBR system will concern the adaptation step. Currently, a mere and not very efficient method is implemented. This method gives good results only when the retrieve source cases are very close to the target problem. For this CBR step, there is three traditional categories:

- Reinstantiation: The solution of the case retrieved is directly
1- used without modification. This strategy is used when the similarity between both cases reaches a very high threshold.
- Substitution: Some values of the retrieved solution attributes
2- are replaced because they are not valid: in conflict or in contradiction with the new problem requirements.
- Transformation: The whole or a part of the retrieved solution
3- must be changed by taking into account some constraints and/or characteristics of the required solution.

The latter two categories need some additional and predefined expert knowledge or heuristics. In the example presented in part 4, the feature material was adapted with a specific rule on the domain. This is one way to proceed, because there are different methods available to capture expert knowledge; rules based methods, constraints satisfaction problem method. The former has problems for maintaining rules (due to this issue, this method is currently less implemented), unlike the latter which opens new possibilities for case adaptation.

Indeed, some additional constraints could be added to improve the quality and accuracy of the proposed solution: user point of view, economical, technical and environmental constraints. Consequently, constraint satisfaction problem methods outperform the performances of the classical and traditional adaptation methods, and it could be an interesting method to implement.

Appendix A

In this appendix we give the demonstration for the test presented in Eq. (14).

Fig. 12 shows the situation in two dimensions. Cells are rejected if:

$$d_t > d_s \tag{A1}$$

$$\Leftrightarrow d_t^2 > d_s^2 \text{ (since both of them are positive)} \tag{A2}$$

But

$$d_t^2 = \sum_{k=x,y}(\xi_k + d_k)^2 \tag{A3}$$

$$d_s = \lfloor d_s \rfloor + \{d_s\} \tag{A4}$$

With $\lfloor d_s \rfloor = \text{floor}(d_s)$ and $\{d_s\} = \text{fractional part of } d_s$

Thus: (A2)

$$\Leftrightarrow \sum_{k=x,y}(\xi_k + d_k)^2 > \left(\lfloor d_s \rfloor + \{d_s\}\right)^2 \tag{A5}$$

$$\Leftrightarrow d_c^2 + \xi^2 + 2*\sum_{k=x,y}\xi_k d_k > \lfloor d_s \rfloor^2 + 2*\lfloor d_s \rfloor\{d_s\} + \{d_s\}^2 \tag{A6}$$

Cells that satisfy (A6) have to be excluded, it is the true condition.

Now, starting with d_t^2 and more precisely with $d_t^2 = d_c^2 + \xi^2 + 2*\sum_{k=x,y}\xi_k d_k$, let us assume that: $\{d_s\} < \xi_k$ whatever k and that

$$d_c > \lfloor d_s \rfloor \text{ Assumptions} \tag{A7}$$

Thus:

$$d_t^2 = d_c^2 + \xi^2 + 2*\sum_{k=x,y}\xi_k d_k > \lfloor d_s \rfloor^2 + \xi^2 + 2*\sum_{k=x,y}\xi_k d_k \tag{A8}$$

$$\Leftrightarrow d_c^2 + \xi^2 + 2*\sum_{k=x,y}\xi_k d_k > \lfloor d_s \rfloor^2 + 2*\{d_s\}^2 + 2*\{d_s\}\sum_{k=x,y} d_k \tag{A9}$$

Because of the triangular relation $\sum_{k=x,y} d_k \geq d_c$, and with the previous assumption $d_c > \lfloor d_s \rfloor$ therefore $\sum_{k=x,y} d_k \geq \lfloor d_s \rfloor$

Consequently:

$$d_c^2 + \xi^2 + 2*\sum_{k=x,y}\xi_k d_k > \lfloor d_s \rfloor^2 + 2*\{d_s\}^2 + 2*\{d_s\}\lfloor d_s \rfloor \tag{A10}$$

Finally, Eq. (A10) demonstrates that the set of chosen assumptions (A7) satisfies the true condition (A6), and cells that satisfy these assumptions can be rejected.

References

Aamodt, A., Plaza, E., 1994. Case-based reasoning: foundation issues, methodological variations and system approaches. Artif. Intell. Commun. 7, 39–59.

Aarts, R., Rousu, J., 1996. Adaptation cost as a criterion for solution evaluation, Smith and Falting Eds., Advanced in Case Based Reasoning, Third European Workshop. EWCBR-96, Lausanne Suisse, vol 1168. Springer-Verlag, Berlin 354-361.

Allen, B.P., 1994. Case-based reasoning: business applications. Communications of the ACM 37 (3), 40–42.

Anand, S., Patterson, D., Hughes, J., 1998. Knowledge intensive exception spaces. In: Proceedings of the 14th Conference on Artificial Intelligence, pp. 574–579.

Avramenko, Y., Kraslawski, A., 2005. Decision supporting system for pre-selection of column internals in reactive distillation. Chem. Eng. Proc. 44, 609–616.

Avramenko, Y., Nystrom, L., Kraslawski, A., 2004. Selection of internals for reactive distillation column-case based reasoning approach. Comp. Chem. Eng. 28, 37–44.

Barioni, M.C.N., Razente, H.L., Traina, A.J.T., Traina Jr, C., 2008. Accelerating k-medoid based algorithms through metric access methods. J. Syst. Software 82, 343–355.

Berchtold, S., Keim, D., Kriegel, H.P., 1996. The X-tree: an index structure for high dimensional data. In: Proceedings of the 22th International Conference on Very Large Data Bases, San Francisco, pp. 28–39.

Bergmann, R., Kolodner, J., Plaza, E., 2006. Representation in case-based reasoning. Knowl. Eng. Rev. 20 (3), 209–213.

Brodu, N., in press. Query sphere indexing for neighborhood requests. J. Graphics Tools.

Bueno, R., Traina, A.J.M., Traina Jr, C., 2007. Genetic algorithms for appromitate similarity queries. Data Knowl. Eng. 62 (3), 459–482.

Chandrasekaran, B., 1990. Design problem solving: a task analysis. Artif. Intell. Mag. 11 (4), 59–71.

Ciaccia, P., Patella, M., Zezula, P., 1997. M-tree: an efficient access method for similarity search in metric spaces. In: International Conference on Very Large Data Bases, Athens, pp. 426–435.

Cortes Robles, G., Negny, S., Le Lann, J.M., 2009. Case based reasoning and TRIZ: a coupling for innovative conception in chemical engineering. Chem. Eng. Process. 48 (1), 239–249.

Deangdej, J., Lukose, D., Tsui, E., Beinat, P., Prophet, L., 1996. Dynamically creating indices for two million cases: a real world problem, Smith and Falting (Eds.), Advanced in Case Based Reasoning, Third European Workshop, EWCBR-96, Lausanne Suisse, Springer-Verlag, Berlin, vol 1168, pp. 105–119.

Finnie, G., Sun, Z., 2003. R^5 model for case-based reasoning. Know. Based Syst. 16, 59–65.

Fox, S., Leake, D., 1995. Using introspective reasoning to refine indexing. In: Proceedings of the 14th Joint Conference on Artificial Intelligence, Montreal, Canada, pp. 391–397.

Galushka, M., Patterson, D., 2006. Intelligent index selection for case based reasoning. Knowl. Based Syst. 19, 625–638.

Giachetti, R.E., Young, R.E., Roggatz, A., Eversheim, W., Perrone, G., 1997. A methodology for the reduction of imprecision in the engineering process, 100, pp. 277–292.

Hjaltason, G.R., Samet, H., 2003. Index driven similarity search in metric spaces. ACM Trans. Database Syst. 28 (4), 517–580.

Hunt, J., 1995. Evolutionary case based design. In: Waston, I.D. (Ed.), Progress in Case-based Reasoning, LNAI 1020. Springer, Berlin, pp. 17–31.

King, J.M.P., Banares Alcantara, R., Manan, Z.A., 1999. Minimising environmental impact using CBR: an azeotropic distillation case study. Environ. Model. Software 14, 359–366.
Kolodner, J., 1993. Case-Based Reasoning. Morgan Kaufmann Publishers, Inc..
Kraslawski, A., Koiranen, T., Nyström, L., 1995. Case-based reasoning system for mixing equipment selection. Comp. Chem. Eng. 19S (S1), S821–S826.
Leake, D.B., Kinley, A., Wilson, D., 1997. Case Based Assessment: estimating adaptability from experience, Fourteenth National Conference on Artificial Intelligence. AAAI press, Menlo Park, CA 674-679.
Lieber, J., 1999. Reformulations and adaptation decomposition. In: Proceedings of the International Conference on Case based reasoning, ICCBR'99, LSA, University of Kaiserslautern, Munich, Germany.
Lieber, J., Bey, P., Boisson, F., Bresson, B., Falzon, P., Lesur, A. Napoli, A., Rios, M., Sauvagnac, C., 2001. Acquisition et modélisation de connaissances d'adaptation, une étude pour le traitement du cancer du sein, Journée Ingénierie des connaissances, IC'2001, Presses Universitaires de Grenoble, Grenoble France, pp. 409–426.
Lieber, J., Napoli, A., 1996. Using classification in case based planning. In: Washlster (Ed.), 12th European Conference on Artificial Intelligence. Wiley and Sons, New York, pp. 132–136.
Lin, K., Jagadish, H., Faloutsos, C., 1994. The TV-tree: an index structure for high dimensional data. VLDB J. 3 (4), 517–542.
Lopez-Arevalo, I., Banares-Alcantara, R., Aldea, A., Rodriguez-Martinez, A., Jimenez, L., 2007. Generation of process alternatives using abstract models and case based reasoning. Comp. Chem. Eng. 31, 902–918.
Malek, M., 1995. A connectionist indexing approach for CBR systems, ICCBR-95, Portugal, pp. 520–527.
Malek, M., 2000. Hybrid approaches for integrating neural networks and case based reasoning: from loosely coupled to tightly coupled models. Soft Comput. Case Based Reasoning, 73–94.
Mille, A., Herbeaux, O., 2007. Accelere: système d'aide à la conception de caoutchouc cellulaire exploitant la remémoration d'expérience, Raisonnement à partir de cas 1, Informatique et Systèmes d'Information. Hermes, Lavoisier, Paris, pp 33-63.
Mori, H., Ibuki, R., Taguchi, K., Futuma, K., Olujic, Z., 2006. Three-component distillation using structured packing: performance evaluation and model validation. Chem. Eng. Sci. 61 (6), 1760–1766.
Negny, S., Le Lann, J.M., 2008. Case-based reasoning for chemical engineering design. Chem. Eng. Res. Design 86, 648–658.
Pal, S.K., Shiu, S.C.K., 2004. Foundations of Soft Case-Based Reasoning. Wiley Interscience, New Jersey.
Patterson, D., Rooney, N., Galushka, M., 2002a. Efficient similarity determination and case construction techniques for case-based reasoning. In: Proceedings of the Fourth European Conference on CBR, pp. 292–305.
Patterson, D., Rooney, N., Galushka, M., 2002b. Towards dynamic maintenance of retrieval knowledge in CBR, Fifteenth International FLAIRS Conference. AAAI Press.
Pralus, M., Geneste, L., 2007. Recherche et adaptation d'expériences structures, imprécises et incomplétes: application en configuration experte, Raisonnement à partir de cas 1, Informatique et Systèmes d'Information, Hermes, Lavoisier, Paris, pp. 65–93.
Richter, M.M., Aamodt, A., 2006. Case-based reasoning foundations. Knowl. Eng. Rev. 20 (3), 203–207.
Schank, R., 1982. Dynamic Memory: A Theory of Learning in Computers and People. Cambridge University Press.
Seuranen, T., Hurme, M., Pajula, E., 2005. Synthesis of separation processes by case-based reasoning. Comp. Chem. Eng. 29, 1473–1482.
Smyth, B., Kean, M.T., 1998. Adaptation-guided retrieval: questioning the similarity assumption in reasoning. Artif. Intell. 102 (2), 249–293.
Stahl, A., 2004. Learning of knowledge intensive similarity measures in case based reasoning, dissertation.de, 986.
Stahl, A., Gabel T., 2003. Using evolution programs to learn local similarity measures. In: Proceedings of the Fifth International Conference on CBR (ICCBR), pp. 537–551.
Stahl, A., Gabel, T., 2006. Optimizing similarity assessment in case based reasoning. Am. Assoc. Artif. Intell., 1667–1670.
Suh, N.P., 1990. The Principles of Design. Oxford University Press.
Surma, J., Braunschweig, B., 1996. Case-base retrieval in process engineering: supporting design by reusing flowsheets. Eng. Appl. Artif. Intell. 9 (4), 385–391.
Traina Jr., C., Traina, A.J.M., Faloutos, C., Seeger, B., 2000. Fast indexing and visualization of metric datasets using Slim-trees. Lect. Note Comput. Sci. 1777, 51–65.
Vieira, M.R., Traina, C., Chino, F.J.T., Traina, A.J.M., 2004. DBM-tree: a dynamic metric access method sensitive to local density data. Brazilian Symp. Databases, 163–177.
Yager, R.R., 1992. On the specificity of a possibility distribution. Fuzzy Sets Syst. 50, 279–292.
Zadeh, L.A., 1965. Fuzzy sets. Inf. Control 8, 338–353.